OSNOVE PROGRAMIRANJA I

1. letnik VSŠ Računalništvo in informatika
Študijsko leto 2005/06

1. Trikotnik števil (24.-28. oktober 2005)

Napišite program, ki najprej prebere število vrstic `n` (`n` je pozitivno celo število), nato pa s pomočjo gnezdenja zank izpiše števila v trikotni obliki tako, kot prikazujejo naslednji primeri:

a) `n=1`
```
1
```

b) `n=5`
```
    1
   23
  345
 4567
56789
```

c) `n=8`
```
       1
      23
     345
    4567
   56789
  678901
 7890123
89012345
```

2. Praštevila (31. oktober - 4. november 2005)

Pruski matematik Christian Goldbach (1690 - 1764) je pokazal, da lahko vsako sodo število *x,* ki je večje od 2, zapišemo kot vsoto dveh praštevil (npr. 4 je vsota praštevil 2+2, 6 je 3+3, 8 je 3+5, 10 je 3+7 ali 5+5, 12 je 5+7 itd.). Napišite program, ki za vse sode *x* med 4 do 100 izpiše vse pare praštevil na naslednji način:

```
4 = 2 + 2
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7 = 5 + 5
12 = 5 + 7
14 = 3 + 11 = 7 + 7
...
22 = 3 + 19 = 5 + 17 = 11 + 11
...
```

Napotek: Najprej napišite funkcijo `jePrastevilo`, ki za argument `n` vrne vrednost `true`, če je `n` praštevilo, in `false`, če `n` ni praštevilo. To funkcijo nato kličite v zanki za vse sode `x` od 4 do 100.

3. Številske vrste (7.-11. november 2005)

Napišite program za izračun vsote naslednjih številskih vrst:

$$e^x = 1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\frac{x^5}{5!}+\frac{x^6}{6!}+\frac{x^7}{7!}+\ldots \qquad |x|<\infty$$

$$\arccos x = \frac{\pi}{2}-\left[x+\frac{1}{2\cdot 3}x^3+\frac{1\cdot 3}{2\cdot 4\cdot 5}x^5+\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}x^7+\frac{1\cdot 3\cdot 5\cdot 7}{2\cdot 4\cdot 6\cdot 8\cdot 9}x^9+\ldots\right] \qquad |x|<1$$

Napotek: Vsoto številske vrste določite tako, da seštejete vrednosti posameznih členov številske vrste za določen x, pri čemer vsak naslednji člen vsote izračunajte s pomočjo predhodnega člena. S prištevanjem členov vsoti zaključite, ko absolutna vrednost člena pade pod 10^{-5}.

4. Razred Avto (14.-18. november 2005)

Sprogramirajte razred `Avto`, ki predstavlja avtomobil z njegovimi tehničnimi značilnostmi in trenutnim stanjem. Poleg atributa `stevilka`, ki omogoča ločevanje posameznih avtomobilov, so tehnične značilnosti opisane z atributi `najvecjaHitrost` (v km/h), `velikostRezervoarja` (v litrih) in `povprečnaPoraba` (v litrih/100 km). Trenutno stanje avtomobila predstavljajo atributi `avtoVzgan` (tipa boolean), `stevecKm` (npr. 234.142 km), `trenutnaHitrost` (npr. 56.12 km/h) in `kolicinaGoriva` (npr. 14.325 litrov).

Razred naj vsebuje naslednje metode:

- `nastaviStevilko(s)`: nastavi atribut `stevilka` na `s` (recimo 1),
- `nastaviNajvecjoHitrost(h)`: nastavi atribut `najvecjaHitrost` na `h` km/h (recimo 180),
- `nastaviVelRezervoar(l)`: nastavi atribut `velikostRezervoarja` na `l` litrov (recimo 48.0),
- `nastaviPovPorabo(l)`: nastavi atribut `povprecnaPoraba` na `l` litrov/100 km (recimo 7.5),
- `napolniGorivo()`: nastavi atribut `kolicinaGoriva` na vrednost `velikostRezervoarja`,
- `vzgiAvto()`: nastavi atribut `avtoVzgan` na true, pri čemer se porabi 0.05 litra goriva (če ni zadosti goriva se avto ne vžge)
- `ugasniAvto()`: nastavi atribut `avtoVzgan` na false, pri čemer se lahko avto ugasne le, ko je njegova hitrost enaka 0.
- `vozi(t)`: vozi `t` minut s trenutno hitrostjo in vrne koliko poti je v tem času prevozil. Metoda povzroči spremembo atributov `stevecKm` in `kolicinaGoriva`. Če je goriva premalo, naj vožnja poteka tako dolgo, dokler ne zmanjka goriva. Avto lahko vozi le v primeru, da je vžgan (atribut `avtoVzgan` nastavljen na `true`) in ima trenutno hitrost večjo kot 0.
- `spremeniHitrost(h)`: spremeni `trenutnaHitrost` na vrednost `h` km/h. Hitrost avtomobila je med `0` in `najvecjaHitrost`.
- `ustavi()`: spremeni `trenutnaHitrost` na vrednost `0` km/h.
- `najvecjiCas()`: metoda vrne podatek, koliko minut in sekund lahko še vozimo glede na trenutno količino goriva. Trenutno stanje avtomobila se ne spremeni.
- `doseg(l)`: metoda vrne podatek, koliko kilometrov lahko prevozimo z največjo hitrostjo, če je v rezervoarju še l litrov goriva. Trenutno stanje avtomobila se ne spremeni.
- `casVoznje(d)`: metoda nam pove vrne koliko časa (v minutah in sekundah) potrebujemo, da prevozimo `d` kilometrov, če vozimo z nespremenjeno hitrostjo, v rezervoarju pa imamo trenutno količino goriva. Trenutno stanje avtomobila naj se sicer ne spremeni.
- `boljsi(a)`: primerja trenutni avtomobil z avtomobilom `a` in izpiše `stevilko` boljšega avtomobila. Boljši je tisti avtomobil, ki v krajšem času prevozi 1000 km, pri čemer upoštevajte tudi čas polnjenja goriva.
- `izpis()`: izpiše vse atribute avtomobila.

Pri vseh metodah predpostavljamo, da avto pri vseh hitrostih porabi enako količino goriva.

Za generiranje avtomobilov sprogramirajte tri konstruktorje:

- prvi naj bo brez parametrov in zgenerira avtomobil, ki ima vse atribute prazne (vrednost 0 ali `false`)
- drugi naj ima le parameter s in naj zgenerira avtomobil s številko s, ostale vrednosti pa naj bodo prazne.
- tretji naj ima parametre `s`, `h`, `v` in `p`, zgenerira pa naj avtomobil številke `s`, z največjo hitrostjo `h`, velikostjo rezervoarja `v` in povprečno porabo `p`. Atributi `stevecKm`, `trenutnaHitrost` in `kolicinaGoriva` naj imajo vrednost 0, `avtoVzgan` pa na `false`.

Za testiranje razreda Avto napišite razred `TestAvto`. Z uporabo vsakega konstruktorja definirajte po en avto. S pomočjo metod za nastavljanje atributov dopolnite podatke za prva dva avtomobila, nato pa izpišite vse podatke o vseh avtomobilih. Z vsemi simulirajte polnjenje goriva, vožnjo, spreminjanje hitrosti in vse informativne izpise. Ponovno izpišite vse avtomobile ter jih primerjajte med seboj.

5. Trgovine (21.-25. november 2005)

V podjetju, ki ima 7 trgovin, spremljajo prodajo po posameznih trgovinah za obdobje petih let. Podatki o prodaji (tj. vrednost vseh prodanih izdelkov v milionih SIT) so shranjeni v dvodimenzionalni tabeli `prodaja`, ki ima 7 vrstic (vsaka vrstica ustreza eni trgovini) in 5 stolpcev (vsak stolpec ustreza enemu letu).

Napišite razred Trgovine, v katerem je tabela prodaja deklarirana kot statična spremenljivka razreda:

```
private static int[][] prodaja = new int[7][5];
```

- Razred naj vsebuje metodo `static void napolniProdajo()`, ki z uporabo generatorja naključnih števil napolni tabelo z vrednostmi med 10 in 1000.

- Napišite metodo `static void izpisiVsoProdajo()`, ki za vsako trgovino posebej izpiše: vrednost prodanih izdelkov v vsakem letu, nato pa še vsoto vrednosti prodaje trgovine v vseh petih letih skupaj.
- Napišite metodo `static void letnaPovprecja()`, ki za vsako leto izpiše vsoto in povprečno vrednost prodaje vseh trgovin skupaj.
- Napišite metodo `static int najTrgovinaLeta(int leto)`, ki vrne indeks trgovine, ki je v letu z indeksom `leto` imela največjo prodajo. Če je takih trgovin več, vrnite indeks prve izmed njih.
- Napišite metodo `static void najProdajaLeta()`, ki izpiše indeks trgovine in indeks (prvega) leta, v katerem je bila dosežena največja prodaja nasploh. Rešitev naj uporablja metodo `najTrgovinaLeta()`.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Pri zadnjih dveh metodah poskrbite za izpis vrnjenih vrednosti, pri predzadnji pa še za ustrezen vnos podatkov (indeks leta).

6. Temperature (28. november - 2. december 2005)

Napišite razred Temperature, ki omogoča statistično obdelavo podatkov o povprečnih mesečnih temperaturah zadnjih desetih let v osmih slovenskih krajih. Podatke hranimo v tridimenzionalni tabeli, kjer prva dimenzija predstavlja 8 krajev, druga 10 let in tretja 12 mesecev vsakega leta.

- Napišite metodo `static void napolniTemperature(double[][][] t)`, ki z uporabo generatorja naključnih števil napolni tabelo prispevki z ustreznimi decimalnimimi števili. Da bi bile generirane vrednosti kar najbolj realne upoštevajte, da so povprečne temperature:
    - pozimi (december, januar in februar) med -20.0 in +5.0 C;
    - spomladi in jeseni (marec, april in maj ter september, oktober in november) med 0.0 in 25.0 C;
    - poleti (junij, julij in avgust) med 20.0 in 35.0 C;

- Napišite metodo `static void izpisiKrajevneTemperature(double[][][] t, int k, int l)`, ki za kraj z indeksom `k` izpiše za leto `l` vse mesečne temperature in celoletno povprečje (celoletno povprečje dobimo tako, da seštejemo povprečne temperature po posameznih mesecih in jih delimo z 12).
- Napišite metodo `static void izpisiTemperaturePoLetih(double[][][] t)`, ki za vsako leto izpiše celoletno povprečje za vsak kraj posebej in za vse kraje skupaj (povprečna vrednost vseh temperatur v vseh mesecih in krajih določenega leta).
- Napišite metodo `static void dveNajnizjiTemperaturi(double[][][] t)`, ki izpiše dve kombinaciji indeksov kraja, leta in meseca, kjer sta bili doseženi dve najnižji mesečni temperaturi nasploh. V primeru, da je več enakovrednih rezultatov, zadostuje izpis prvega.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Upoštevajte, da tabela temperatur **ni** statična spremenljivka razreda. V metodi `main` poskrbite tudi za ustrezen vnos podatkov.

7. Analiza nizov (5. – 9. december 2005)

Napišite razred `AnalizaNizov`, ki se obnaša kot knjižnica metod, s katerimi lahko analiziramo vsebino poljubnega niza. Razred naj vsebuje naslednje metode:

- Metoda `public static int steviloMalihCrk(String niz)` vrne število malih črk, ki jih vsebuje `niz`.
- Metoda `public static int steviloZnakov(String niz, char c)` vrne število pojavitev znaka `c` v nizu `niz`.
- Metoda `public static int steviloBesed(String niz)` vrne število besed, ki jih vsebuje `niz`. Pri tem predpostavite, da se lahko vsaka beseda začne s črko ali številko (in vsebuje poljubno število črk in številk), ločilo med besedami pa so beli presledki. Med bele presledke (angleško *white space*) štejemo znak za presledek (`' '`), znak za tabulator (`'\t'`), znak za novo vrstico (`'\n'`), znak za prehod na začetek vrstice (angl. *carriage return*, `'\r'`) ter znak za pomik v novo vrsto (angl. *line feed*, `'\f'`).
- Metoda `public static int steviloEnakihZnakov(String niz)` vrne največje število enakih (katerihkoli) malih črk (angleške abecede) v nizu `niz`.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Pri tem poskrbite za ustrezen vnos niza in drugih potrebnih podatkov.

8. Kodiranje (12. - 16. december 2005)

Napišite razred `Kodiranje`, ki vsebuje metodi

```
public static StringBuffer kodiraj(String n) in
public static String dekodiraj(StringBuffer n)
```

Metoda `kodiraj` naj niz `n` (ki vsebuje le črke angleške abecede) pretvori v niz tipa `StringBuffer` tako, da vsako črko angleške abecede nadomesti z nizom, ki predstavlja zaporedno številko velike črke v angleški abecedi. Da bi omogočili enolično dekodiranje, je treba 'A' in 'a' nadomestiti z "01" ter 'B' in 'b' z "02", medtem ko ostale črke lahko nadomestimo z eno ali dvema ciframa ('C' in 'c' s "3", 'D' in 'd' s "4", ... ter 'Z' in 'z' s "26").

Metoda `dekodiraj` pa naj tako zakodiran niz pretvori spet v prvotno obliko. V primeru klica metode `dekodiraj` s pokvarjenim zakodiranim nizom, naj ta metoda ugotovi napako (primer pokvarjenega niza je recimo niz "03").

Primer: Metoda `kodiraj` pretvori niz "RACUNALNIK" v "18013211401121491 1", metoda dekodiraj pa ta niz pretvori ponovno v "RACUNALNIK".

Delovanje obeh metod preverite tako, da v metodi `main` zaporedoma vnašate (poljubno število) nizov, za katere program najprej izpiše njihovo kodo, nato pa še niz, ki nastane z dekodiranjem kode. Postopek kodiranja naj se zaključi takrat, ko vnesete prazen niz.

9. Planinci (19. – 23. december 2005)

Razred `Planinec` je namenjen hranjenju podatkov o članih planinskega društva. Obsega številko, priimek in ime planinca, leto prvega vpisa in dvodimenzionalno tabelo, v kateri so v časovnem zaporedju shranjeni podatki o njihovih planinskih podvigih v Sloveniji: prva dimenzija predstavlja zaporedna leta od vpisa, druga število osvojenih vrhov, vrednost sama pa višino določenega vrha v metrih. Ker je najstarejši planinec vpisan v društvu 30 let, je 30 tudi velikost prve dimenzije, medtem ko je število osvojenih vrhov v določenem letu omejeno na 20.

```
class Planinec
{
  private int stevilka;
  private String ime,priimek;
  private int letoVpisa;
  private int[][] podvigi = new int[30][20];
}
```

Primer:
```
stevilka: 4
ime: Janez
priimek: Novak
letoVpisa: 2004
podvigi = {{890,1250,2060,2369,1024,2864,1028},{1290,1462}}
```

Sprogramirajte razred `TestPlaninec`, v katerem naj bo deklarirana tabela s podatki za 10 planincev:

```
class TestPlaninec
{
  final int ST_PL=10;
  Planinec[] planinci = new Planinec[ST_PL];
}
```

Razred `TestPlaninec` naj:

- omogoča vnos oziroma naključno generiranje podatkov v tabeli `planinci`. Številka planinca naj bo kar za ena povečan indeks iz tabele `planinci`, priimek in ime naj se vneseta ročno, medtem ko se ostali podatki generirajo naključno:
  - leto prvega vpisa kot naključna vrednost med 1976 in 2005
  - število osvojenih vrhov kot naključna vrednost med 0 in 20
  - višine posameznih vrhov kot naključna vrednost med 800 in 2864.
- za vsakega planinca urejeno izpiše podatke o vseh njegovih podvigih (po letih), skupaj z izpisom števila osvojenih vrhov skupaj

Primer:
```
4 Janez Novak, leto vpisa: 2004, skupaj vrhov: 9
2004 : 890 m, 1250 m, 2060 m, 2369 m, 1024 m, 2864 m,1028 m
2005 : 1290 m, 1462 m
```

- za vsako leto (od 1976 do 2005) izpiše število vseh osvojenih vrhov (vseh planincev)
- izpiše urejen seznam planincev glede na najvišji osvojeni vrh. Na prvem mestu naj bo planinec, ki je osvojil najvišji vrh, na zadnjem pa tisti, ki je osvojil najnižjega (oziroma nobenega).

10. Funkcije (2.-6. januar 2006)

Abstraktni razred `Funkcija` predstavlja osnovo za različne razrede, s katerimi realiziramo različne matematične funkcije ene spremenljivke *f(*x):

```
public abstract class Funkcija
{
  public abstract double vrednost(double x);
  public double enaNicla(double x1, double x2)
  {
    // to metodo morate sprogramirati
  }
}
```

Metoda `vrednost` izračuna **vrednost funkcije** v točki `x` – *f(x)*, metoda `enaNicla` pa **eno ničlo** (vrednost `x`, kjer je vrednost `f(x)=0`), ki se nahaja med točkama `x1` in `x2`.
Dopolnite zgornjo deklaracijo razreda `Funkcija` z metodo `enaNicla`, ki poišče (po metodi bisekcije) eno izmed ničel, ki se nahajajo med točkama `x1` in `x2` na naslednji način:

a. Parametra `x1` in `x2` morata biti izbrana tako, da je v eni izmed teh dveh točk funkcijska vrednost pozitivna, v drugi pa negativna. V kolikor ni tako, ugotovite napako.
b. Interval `(x1,x2)` razpolovimo tako, da izračunamo `x=(x1+x2)/2`.
c. Izračunano srednjo vrednost uporabimo namesto enega od prejšnjih približkov:
   - Če imata vrednosti polinoma v točkah `x` in `x1` enak predznak, nadaljujemo z iskanjem ničle na intervalu `(x,x2)`, torej `x1=x;`
   - V nasprotnem primeru nadaljujemo z iskanjem ničle na intervalu `(x1,x)`, torej `x2=x.`
d. Koraka (ii) in (iii) ponavljamo, dokler ne dosežemo predpisane natančnosti (dokler velikost intervala ne pade pod 10^{-4}).

- Napišite razred `Polinom`, ki je podrazred razreda `Funkcija` in služi za predstavitev polinomov. Vsak objekt tega razreda naj bo definiran s tabelo realnih števil `a`, ki predstavljajo koeficiente polinoma:

$$p(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-1} x + a_n$$

  V razredu `Polinom` sprogramirajte ustrezen konstruktor in redefinirajte metodo `vrednost`, tako da vrne vrednost polinoma v točki x. Za izračun vrednosti polinoma uporabite Hornerjevo shemo, po kateri lahko polinom zapišemo na naslednji način

$$p(x) = (\ldots((a_0 x + a_1)x + a_2)x + \ldots + a_{n-1})x + a_n$$

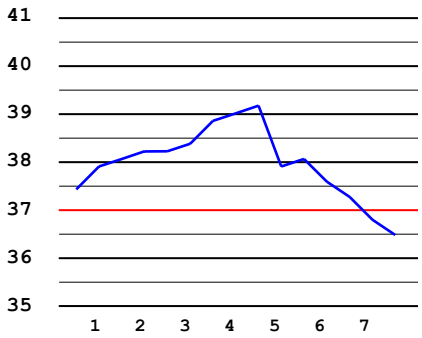
- Napišite testni razred `TestFun`, v katerem v metodi `main` s pomočjo generatorja naključnih števil tvorite 10 polinomov. Za vsako od njih izpišite njihove ničle na ustreznih intervalih.

11. Grafični prikaz temperature pacienta (9.-13. januar 2006)

Napišite program, ki bo izrisal graf temperature pacienta v zadnjem tednu z lomljeno črto. Predpostavite, da temperaturo merijo dvakrat dnevno (ob 6.00 in ob 18.00 uri), izmerjene vrednosti pa so shranjene v dvodimenzionalni tabeli `t` , ki že obstaja kot globalna spremenljivka:

```
static double[][] t={{37.5,37.9},{38.1,38.2}, ... , {37.1,36.7}};
```

Diagram naj se izriše v ustrezno velikem oknu (velikost je odvisna od velikosti zaslona). Program naj najprej nariše vodoravne črte, ki predstavljajo temperature med 35 in 42 stopnj (na vsake pol stopinje), pri čemer naj bo črta ob 37 rdeče barve, ostale pa črne, ob celih stopinjah pa naj bodo črte debelejše. Pred črtami naj se na ustreznih mestih izpišejo stopinje, pod črtami pa zaporedna številka dneva. Enako kot na sliki naj se gibanje temperature pacienta prikaže z debelejšo modro lomljeno črto, ki predstavlja obdobja od meritve do meritve.



A*. Veriga števil (21.-25. november 2005)

Podan je trikotnik števil z `N` vrsticami (`N` < 100) , pri čemer je v i-ti vrstici i števil (glej sliko). Trikotnik naj sestavljajo poljubna cela števila z vrednostmi `x` med `A` in `B` (spodnjo in zgornjo mejo) .

Primer trikotnika za N=6, A=0 in B=12:

```
          9
        8   2
      6   8   3
    4   9   5  11
  9   8   7   5   8
12  11  0   6   1  11
```

Napišite čimbolj učinkovit program, ki najprej prebere ustrezna števila `N`, `A` in `B` ter zgenerira trikotnik naključnih števil. Program naj izračuna **največjo možno vsoto** števil v verigi, ki se prične pri korenu in konča pri kateremkoli številu zadnje vrstice trikotnika. Pri tem velja, da sta veljavna naslednika nekega števila samo števili, ki sta neposredno pod njim (npr. za število 2 sta to števili 8 in 3).

Program naj najprej izpiše vsoto števil verige, nato pa še verigo števil (od vrha trikotnika do zadnje vrstice), ki smo jih sešteli skupaj. **Tudi če je takšnih verig več, zadostuje izpis ene izmed njih.**

Primer: Za gornji trikotnik je pravilni rezultat:

```
Vsota verige: 53
Veriga števil: 9, 8, 8, 9, 8 in 11
```

B*. Planinci 2 (19. – 23. december 2005)

Pri reševanju naloge 9 upoštevajte še naslednje:

- dvodimenzionalna tabela planinskih podvigov naj bo prilagojena vsakemu planincu posebej: velikost prve dimenzije, ki predstavlja zaporedna leta članstva od vpisa do danes, naj tako **ne bo** vedno 30, temveč naj bo enaka številu let članstva vsakega planinca. Enako naj velikost druge dimenzije predstavlja število osvojenih vrhov izbranega planinca v določenem letu (pri generiranju podatkov to število določite naključno). Upoštevajte, da so v tabeli samo planinci, ki imajo vsaj en podvig.

- dodajte metodo za dodajanje novega podviga v določenem letu. Upoštevajte, da pri tem postane tabela podvigov premajhna in jo je treba ustrezno povečati.

- dodajte metodo za brisanje podviga, ki je pri izbranem planincu na določenem mestu v določenem letu. Upoštevajte, da pri tem postane tabela podvigov prevelika, zato jo je treba pomanjšati. Upoštevajte tudi, da mora imeti vsak planinec vsaj en podvig, zato zadnjega podviga ni mogoče brisati.

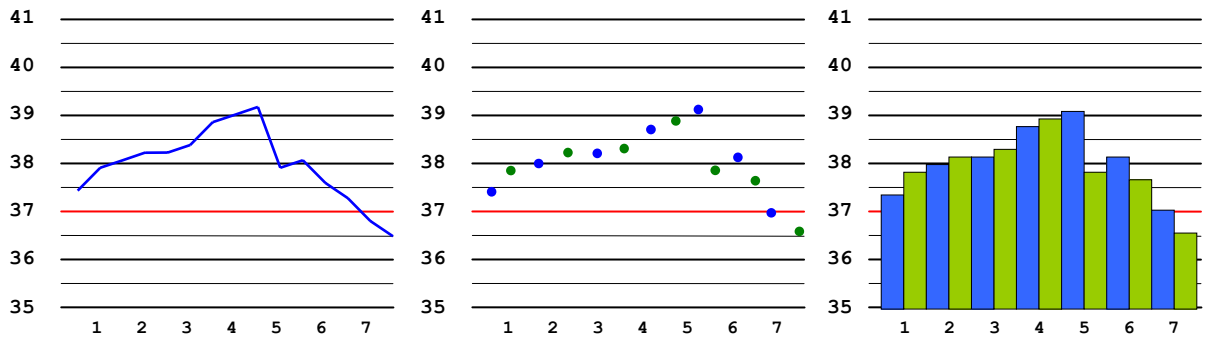
- metodo za urejen izpis planincev spremenite tako, da se upošteva samo izbrano obdobje v koledarskih letih (recimo od 1996 do 2001).

C*. Grafični prikaz temperature pacienta 2 (9.-13. januar 2006)

Nadgradite rešitev naloge 11 tako, da bo omogočen izris grafa temperature pacienta na tri načine:

- z lomljeno črto (zahteva naloge 11),
- s točkami in
- s stolpci.

Prvi način povezuje vrednosti posameznih meritev temperature pacienta z modro lomljeno črto, drugi na ustreznih mestih izrisuje točke (jutranje modre, večerne pa zelene), tretji pa izrisuje ustrezno visoke stolpce, ki predstavljajo obdobja od meritve (recimo ob 6.00) do meritve (ob 18.00) - za jutranjo meritev naj bodo stopci modri, za večerno pa zeleni (glej sliko). Preklapljanje med tremi načini izrisa naj bo s pomočjo gumba v oknu: prvi klik preklopi iz prvega načina v drugega, naslednji iz drugega v tretjega, še naslednji iz tretjega v prvega itd.



NAVODILA:

- Naloge lahko delate doma, vendar jih morate zagovarjati na laboratorijskih vajah do predpisanih rokov (ti so navedeni ob vsaki nalogi). V kolikor vaje na določen rok odpadejo (prazniki), morate naloge zagovarjati na naslednjih vajah.
- Striktno se držite razporeda laboratorijskih vaj.
- Študenti, ki iz upravičenih razlogov (na podlagi pisnega dokazila o bolezni ipd.) ne bodo pravočasno zagovarjali naloge, morajo to storiti na prvem naslednjem roku.

- V zimskem izpitnem obdobju (januar in februar 2006) bodo lahko opravljali izpit samo študenti, ki bodo pravočasno opravili vse obvezne naloge.
- V poletnem izpitnem obdobju (junij 2006) bodo lahko opravljali izpit študenti, ki bodo pravočasno opravili vsaj 7 obveznih nalog.
- Ostali študenti bodo lahko šli prvič na izpit šele v jesenskem izpitnem obdobju (septembra 2006).
- Naloge A*, B* in C* so težje naloge in niso obvezne. Študenti, ki bodo opravili te naloge do predpisanega roka, bodo dobili za vsako nalogo (največ) 5 dodatnih točk. Te se bodo upoštevale (samo) pri prvem polaganju pisnega izpita v zimskem izpitnem obdobju.
- Naloge niso obvezne za študente, ki so ponovno vpisani v 1. letnik. Narediti jih morajo le, če želijo pridobiti dodatne točke s pomočjo nalog A*, B* in C* (v tem primeru morajo zagovarjati tudi vse obvezne naloge).
- Študent, ki zamudi rok za zagovor obvezne naloge, lahko namesto te naloge naredi eno izmed dodatnih nalog (tako, ki ji še ni potekel rok za oddajo), vendar se v tem primeru ne upoštevajo dodatne